art news

Kyle J. Knoepfel

26 October 2017

Items for discussion

- *art* 2.08.04 and 2.09.01
- Anticipating *art* 2.10

art 2.07 → 2.08

- The primary change going from *art* 2.08 to 2.08 was relaxing the restrictions required for processing input files.
- *art* 2.08 and beyond allows in the same job the processing of input files with arbitrarily different processing histories.
- Enabling this facility required changing the schema of the `art::ProductID`, which is the underlying construct used in providing `art::Ptr` and `art::Assns` support.
- Schema evolution rules were put in place to facilitate reading old (i.e. pre-2.08 files)
- For *art* 2.08.00, 2.08.01, 2.08.02, **2.08.03** and **2.09.00**, the schema evolution rules are incorrect.
  - In principle, all `art::Ptrs` and `art::Assns` read using these versions are affected.
  - See following slides for details.
- The schema evolution rules have been fixed for *art* **2.08.04** and **2.09.01**.

Two examples illustrating the error:

- `art::Ptrs` may become null

```
---- ProductNotFound BEGIN
 A request to resolve an art::Ptr to a product containing items of type: std::string with ProductID 0
 cannot be satisfied because the product cannot be found.
 The productGetter was not set -- are you trying to dereference a Ptr during mixing?
 cet::exception going through module PtrmvAnalyzer/ptrmvReader run: 1 subRun: 0 event: 1
---- ProductNotFound END
```

- Using an `art::FindMany(P)` object may result in an empty association list for a given reference element (see redmine issue #17898):

```
auto const& tracks = e.getValidHandle<Tracks>(tag_);
art::FindMany<Hits> hitsForTrack{tracks, e, assnsTag_};

for (std::size_t i{}; i < hitsForTrack.size(); ++i) {
  hits = hitsForTrack.at(i);
  assert(!hits.empty());  // May fail when it used to succeed
}
```

How do I know if I am affected?

- The following criteria must be met to expose the problem:
  - *art* 2.08 or newer must be set up
  - The input file must have been produced by a version of *art* older than 2.08
  - The input file has `art::Ptrs` or `art::Assns` in it.
  - In the processing history of the input file,
    a) there must be at least one process in which no products were produced and no events were filtered; and
    b) that process ***must precede*** another process in which products *were* produced or events were filtered

- Example of an affected process:
  - Process 1 produces products
  - Process 2 is a concatenation job (no new products produced, no filtering)
  - Process 3 produces more products

How do I know if I am affected?

- A workflow that's affected:

Processing step	*art* version	Notes
Generation	2.05	
Concatenation	2.05	no products produced
Simulation	2.06	writes sim.root
Reco 1	2.08.03	reads sim.root, writes reco_1.root
Reco 2	2.08.03	reads reco_1.root

- The incorrect translation was done in the Reco 1 step; that means that `reco_1.root` is potentially unusable.

How do I know if I am affected?

- A workflow that's **not** affected:

Processing step	*art* version	Notes
Generation	2.05	
Simulation	2.06	writes sim.root
Reco 1	2.08.03	reads sim.root, writes reco_*.root
Concatenation	2.08.03	reads reco_*.root writes, reco_combined.root
Reco 2	2.08.03	reads reco_combined.root

- The translation was done correctly in the Reco 1 step; that means that `reco_combined.root` is well-formed.

Going forward

- *art* 2.08.04 and 2.09.01 have corrected schema evolution rules.

- This was a difficult problem because it relates to backwards compatibility of files produced by art:
  - The files used to test the schema evolution rules for *art* 2.08 did not include a processing history that would lead to this issue.

- The *art* team is working toward improving its testing procedures to better catch these problems in the future.

- My apologies.

art series 2.10

- We have been considering various features/changes for *art* 2.10.
- We are intending to provide:
  - GCC 7.2 (-std=c++17) compiler/build of *art* (*art* would not directly use any C++17 features)
  - Clang 5.0 (-std=c++17) compiler/build of *art*
- We are intending to remove:
  - cbegin/cend free functions from cetlib
    - `cet::cbegin(…)` → `std::cbegin(…)`
    - `cet::cend(…)` → `std::cend(…)`
  - `reconfigure()` virtual function from module base classes
    - *art* does not call it
    - Users can still provide their own, but must remove any (`virtual` or) **`override`** keywords that accompany it
- Potential changes to anticipate multithreading (under discussion)